Experiences from the Future - Using Object-Oriented Concepts for 3D Visualization and Validation of Industrial Scenarios

Volker Luckas
Fraunhofer Institute for Computer Graphics (IGD)
and
Ralf Dörner
Fraunhofer Applications Center for Computer Graphics in the Chemical and Pharmaceutical Industry (AGC)

An integrated application framework is described suitable for the automatic generation of 3D visualizations and animation based upon sensor, simulator or layout planning data. The principal contribution is the introduction of the term animation element and the description of the underlying concept. The technical innovation is the object-oriented design of animation elements integrating geometry and individual, object-specific behavior. Animation elements build an object-oriented framework as the elements may be customized for a specific application. Moreover, a framework for implementing appropriate animation and visualization tools has been developed.

Additional Key Words and Phrases: Object-orientation, 3D visualization, Simulation, Animation Element, Automation

1. INTRODUCTION

This paper describes an application framework [Fayad and Schmidt 1997] for the automatic generation of 3D visualizations and animation based upon abstract industrial simulation or layout planning data output.

Therefore a framework is needed capable of generating attractive visualizations that makes use of the provided data. In particular, our approach has to obey certain requirements of the industrial context, i.e. the generation has to be done

Name: Volker Luckas
Address: Rundeturmstraße 6, D-64283 Darmstadt, Germany
Affiliation: Fraunhofer Institute for Computer Graphics (IGD)
Name: Ralf Dörner
Address: Varrentrappstraße 40-42, D-60486 Frankfurt, Germany
Affiliation: Fraunhofer Applications Center for Computer Graphics in the Chemical and Pharmaceutical Industry (AGC)

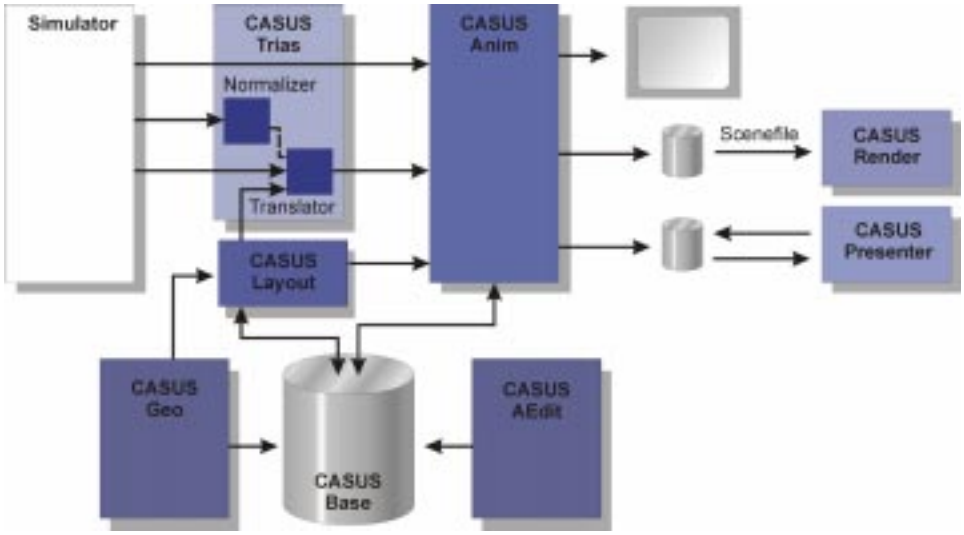

Fig. 1. Main parts of the application framework CASUS System.

(1) with little effort in terms of time and costs
(2) by non-experts
(3) in a flexible, reusable and reliable way.

Since current visualization and animation tools [Mahieddine and Lafon 1990; Gervautz and Beltcheva 1994] do not fulfill these specific requirements, we propose a new approach based upon a consistent usage of object-oriented paradigms [Meyer 1988; Booch 1994] at two levels:

(1) a framework of basic Clipart-like animation elements that rely on an object-oriented data structure
(2) a framework of applications that can be used to select, compose and visualize these animation elements in an appropriate way

The following section deals with the conception of level 2 while the next section shows how object orientation itself can be used as an approach for the definition of 3D visualization and animation (level 1).

2. CONCEPTION OF THE APPLICATION FRAMEWORK

In this section, the process of creating 3D animation using the individual modules that form the proposed application framework is described. Prototypes of the modules have already been implemented in the context of CASUS System [Luckas and Broll 1997]

An overview of the whole system is depicted in figure 1 and described in more detail below.

Starting with an application scenario, the first step to be done is a widespread information gathering process. In the production and logistics context, for example, information, such as machine characteristics, throughput, orientation, location of

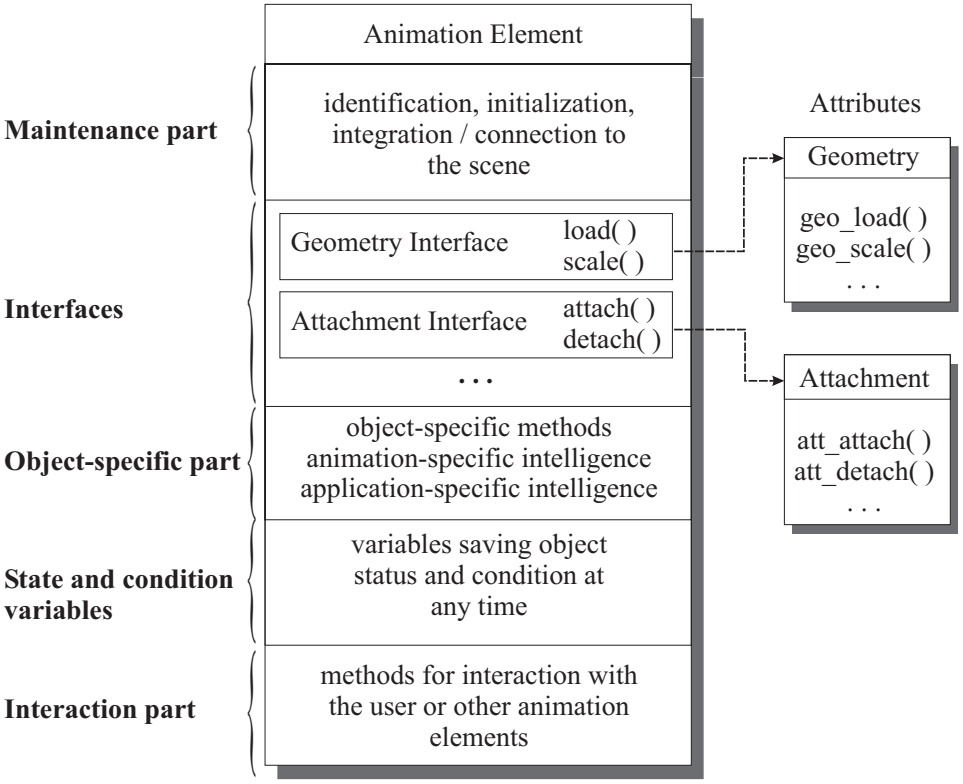

Fig. 2. Structure of an animation element.

each individual object, as well as the figures to be optimized, have to be detected and made available to the whole system. Based upon this information, a simulation model and a 3D visualization model have to be created.

For this task, a layout editor is provided where predefined 3D objects called 'animation elements', taken from a library, can be interactively placed, scaled and oriented. In order to place them on 2D layout plans, the animation elements are represented in this module analogous to 2D Clipart. The elements are stored in an animation elements library called CASUS Base, offering database functionality, especially for searching the appropriate elements. This allows the customer to set up a scaled model of the real scenario within just a short period of time.

Since a realistic representation of the dynamics inside a scenario is based upon simulation results calculated with specific event-oriented simulators, all the elements and components of the simulation model have to be directly mapped to animation elements in the 3D model. This can be done automatically with the module CASUS Trias. Furthermore, it takes the events produced by an simulator, normalizes and translates them to method invocations of the corresponding animation element. Thus, animation elements must not only consist of geometry, but also provide functionality available through object-specific methods. By normalizing simulator traces the CASUS Trias module allows to be adapted for arbitrary

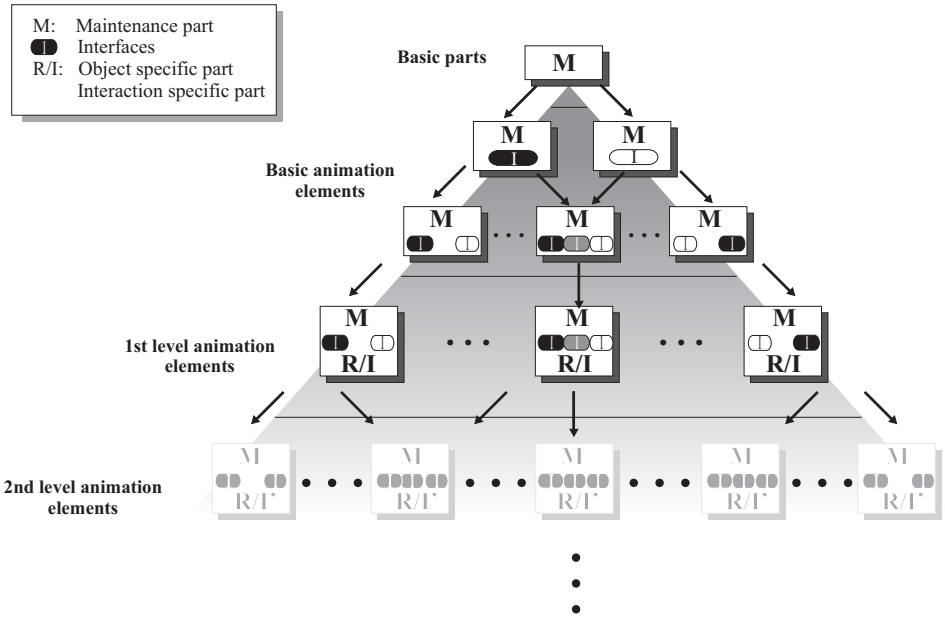

Fig. 3. Hierarchy tree of the animation element concept.

event-oriented simulators.

In the next step, an animation system is needed that can be addressed through the method invocations of the animation elements and that can generate the animation itself. In the corresponding module, CASUS Anim, this can be done in a flexible way. Animation parameters, such as time (frames per second, length of the animation, visible time span) or illumination parameters, can be configured by the customer. Additionally, quality aspects like pixel count or different levels of geometrical abstraction, as well as different output formats, are supported. In order to present the generated 3D animation, visualization modules are provided, such as a renderer (CASUS Render) or a VRML [ISO/IEC DIS 14772-1 1997] browser (CASUS Presenter [Schäfer et al. 1997]). Thus, CASUS System is able to fulfill individual requirements, such as high-end production rendering for marketing purposes or distribution of complete 3D visualizations over the Internet for validating.

Altogether, the system framework for an automatic generation of 3D visualizations consists of six central applications: the (event-oriented) simulator, the animation element library, the layout editor, the simulator trace translator and normalizer, the animation system and the visualization systems.

3. CONCEPTION OF ANIMATION ELEMENTS

As we saw in the previous section, the term 'animation element' plays a central role within the whole concept of the application framework. The animation element concept is the result of transferring the object-oriented paradigm to 3D animation. An animation element can be considered a class. It has attributes like geometry, position, material, sound, state variables, paths, reference points (abstract identifiers for 3D coordinates on the object) and attachments for defining logical hierarchies.

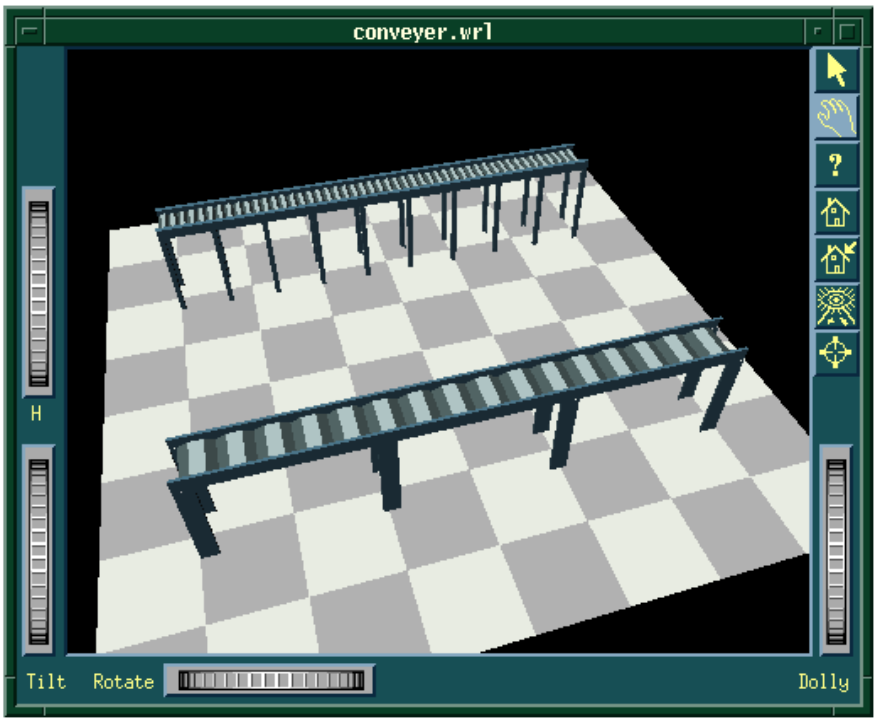

Fig. 4. Object-specific scaling method for a conveyor belt.

All attributes may be combined individually following the aggregation principle defining the animation element's basic complexity. The attributes themselves are represented as classes, offering selected sets of methods representing their typical functionality; for example, the attribute position offers methods for placing and rotating the animation element's geometry. Besides, the animation element itself offers object-specific methods.

All animation element classes are derived directly or implicitly from a set of base classes having one abstract class as root. Thus, they provide a defined interface allowing a uniform access for automated processing. Using the mechanisms of inheritance and overloading, animation element classes are able to provide additional object-specific functionality. The scale method for a conveyor, for instance, has to be redefined in order to avoid geometric distortion. Instead of a simple geometric scale, additional rollers and feet have to be added and scaled (see figure 4). The underlying concept guarantees that existing animation elements may also be tailored to suit special requirements without having to write new animation element classes from scratch.

From each animation element class, an arbitrary number of instances can be created, being independent and distinguished by different states. This may be used, for example, to implement crowd animation in an elegant way.

Being classes, the animation elements also offer the other advantages of object ori-

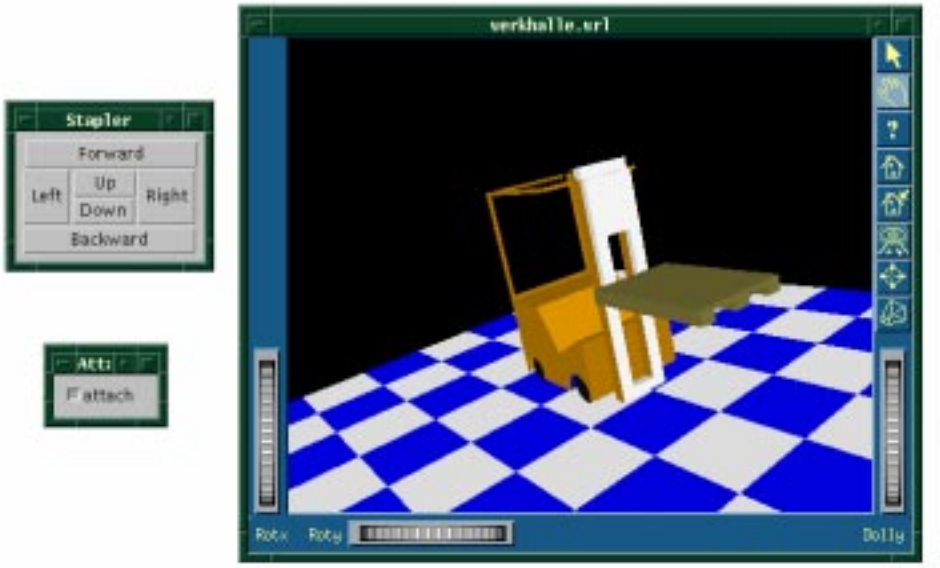

Fig. 5. Animation element 'forklift' with object-specific methods in VRML.

entation, such as encapsulation, modularization or easy maintenance [Meyer 1988].

As shown in figure 1, there is a module provided in the CASUS System called CASUS AEdit, that supports the user when creating or modifying an animation element. Additionally, the module CASUS Geo can be used to convert between different geometric formats that are used for the geometry attribute.

The animation element concept allows the customer to predefine the functionality and behavior as well as the visual appearance of each object being part of a future visualization. With this concept it is possible to generate a visualization just by telling each animation element how to behave within a period of time. The customer is not confronted with animation parameters or visualization problems since they are already provided by the animation element. The integration of geometry and behavior even guarantees the reusability of all animation elements since there no application specific dependency has to be considered.

4. REALIZATION

The modules of the application framework are implemented in C++ [Stroustrup 1991] and Java following an object-oriented conception.

The animation elements are implemented as C++ classes with references to external files where the geometric description is stored. The geometric modeling is done by professional and experienced designers in order to ensure high visual quality. Following this conception, the designers are not required to deal with object-oriented programming. Geometry and class definition are stored together inside a library to guarantee uniform access.

The translator CASUS Trias produces an animation description generating a

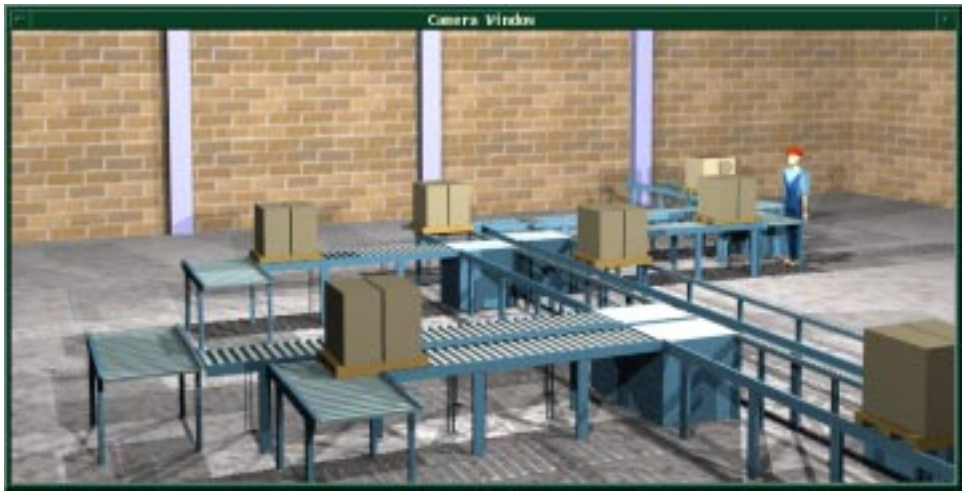

Fig. 6. Visualization example of a production and logistics scenario.

C++ source code. It uses the description of the layout editor for instantiation of objects in the scenario and invocation of methods for placing, orientating and scaling. CASUS Trias is also responsible for the dynamic part of the animation script since it maps simulation events to method calls with corresponding parameters.

This C++ code is compiled and linked to the animation system kernel implemented as a class library. The resulting executable animation contains all information necessary for visualization and is completely controlled by CASUS Anim. The parameters the user can configure in CASUS Anim are send to the executable animation that generates a corresponding visualization , e.g. in a rendering format or in VRML.

Another solution we propose is to implement animation elements in Java and VRML where a Java class is related to a VRML description via the VRML script node. Then, the animation element object can be used 'as is' in a VRML-based animation description. The animation system's task in this case is to build up the VRML scene graph, integrating the animation elements as leaves via the VRML transform node. The advantage to this is that the object-specific functionality of an animation element is still present and may be addressed not only by the creator of the animation description, but by the end users, as well.

In order to validate our realization concepts of an integrated, object-oriented application framework, we used our implemented prototypes in certain industrial scenarios. These were mainly located in production and logistics. We have been able to successfully connect CASUS System to several event-oriented simulators and have generated 3D visualizations for various industrial companies. CASUS System is also used in the context of the Demonstration Center Simulation in Production and Logistics at the Fraunhofer Gesellschaft, Germany [DZ-SIMPROLOG].

CASUS System was tested on different platforms, including IRIX, Solaris and Windows NT.

5. CONCLUSION

In this paper, we presented a flexible, object-oriented application framework for an integrated generation of 3D visualization and animation using predefined animation elements. The paradigm of object orientation has been applied to the modules of the framework, as well as to the animation elements themselves. Thus, with this approach, the goal of a low-cost and fast production by non-experts can be reached.

Besides the possibility of distributing 3D visualization and animation over the Internet, there are several additional advantages, such as 3D navigation support and platform independence, that are especially attractive to the customer. The classical output formats, such as video streams or online presentations, are also supported, since the output format generation is done using generic driver modules.

Flexibility and direct and realistic visualization capabilities establish the basis for a fast, cheap and reliable decision-making process.

REFERENCES

Beeson, G. 1997. An object-oriented approach to VRML development. In *Proceedings of VRML 97* (Monterey, CA, 1997).

Booch, G. 1994. *Object-oriented Analysis and Design with Applications* (2nd ed.). The Benjamin/Cummings Series in Object-oriented Software Engineering. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA.

Dörner, R., Luckas, V., and Spierling, U. 1997. Ubiquitous animation – an element-based concept to make 3D animations commonplace. In L. Pocock, R. Hopkins, D. Ebert, and J. Crow Eds., *Visual Proceedings – The Art and Interdisciplinary Programs of SIGGRAPH 97, COMPUTER GRAPHICS Annual Conference Series* (Los Angeles, CA, Aug. 1997), pp. 210. ACM SIGGRAPH: The Association for Computing Machinery, Inc., New York.

DZ-SIMPROLOG. Demonstration Center Simulation in Production and Logistics (DZ-SIMPROLOG). http://www.igd.fhg.de/dzsim/.

Encarnao, J. L. et al. 1993. Advanced Research and Development Topics in Animation and Scientific Visualization. In R. A. Earnshaw and D. Watson Eds., *Animation and Scientific Visualization, Tools and Applications*, Academic Press, Harcourt Brace & Company.

Fayad, M. and Schmidt, D. 1997. Object-oriented application frameworks. *Communication of the ACM 40*, 10 (October).

Gervautz, M. and Beltcheva, O. 1994. An approach for object-oriented animation design. Technical Report TR-186-2-94-2 (Aug.), Institute for Computer Graphics, Technical University Vienna, Vienna.

ISO/IEC DIS 14772-1. 1997. The virtual reality modeling language, VRML97 specification. ISO/IEC DIS 14772-1 (April).

Luckas, V. and Broll, T. 1997. CASUS – an object-oriented, three-dimensional animation system for event-oriented simulators. In N. Magnenat-Thalmann and D. Thalmann Eds., *Proceedings Computer Animation '97* (Geneva, June 1997), pp. 144–150. University of Geneva (MIRALab, CUI), Swiss Federal Institute of Technology (LIG), Swiss National Research Foundation: IEEE Computer Society Press, Los Alamitos, CA.

Mahieddine, M. and Lafon, J. C. 1990. An object-oriented approach for modelling animated entities. In N. Magnenat-Thalmann and D. Thalmann Eds., *Proceedings of Computer Animation 1990* (1990), pp. 177–187. Springer Verlag, Tokio.

Meyer, B. 1988. *Object-oriented Software Construction*. Prentice Hall, New York, NY.

Schäfer, A., Müller, W., and Luckas, V. 1997. A java based VRML 2.0 browser. In *Poster Proceedings, 6th International World Wide Web Conference* (Santa Clara, CA, July 1997).

Schmidt, D. and Fayad, M. 1997. Lessons learned. *Communication of the ACM 40*, 10 (October).

Schmidt, D., Fayad, M., and Johnson, R. 1996. Software patterns. *Communication of the ACM 39*, 10 (October).

Stroustrup, B. 1991. *The C++ Programming Language* (2nd ed.). Addison-Wesley Publishing Company, Inc., Reading, MA.